UNIVERSIDADE ABERTA
www.uab.pt

Ministério da Ciência, Tecnologia e Ensino Superior

U.C. 21046

Estruturas de dados e algoritmos fundamentais

21 de Junho de 2013

INSTRUÇÕES

PARA A RESOLUÇÃO DO EXAME, ACONSELHA-SE QUE LEIA ATENTAMENTE O SEGUINTE:

1. O exame está dimensionado para um tempo de resolução de duas horas e trinta minutos
2. Este exame cobre toda a matéria da unidade curricular.
3. O exame é constituído por 5 questões.
4. A cotação de cada uma das questões está indicada.
5. O exame deve ser resolvido SEM CONSULTA.
6. O exame deve ser escrito a esferográfica.
7. A interpretação dos enunciados das questões também faz parte da sua resolução, pelo que, se existir alguma ambiguidade, deve indicar claramente como foi resolvida.

Bom Trabalho!

1º Questão (4 Valores)

a) Qual o tempo de execução do seguinte algoritmo entre as seguintes alternativas: log N, N, N*logN, N^2

```
double power(double x, int n)
{
 double res;
 for (res = 1.0;  n-- >0; res*=x);
 return res;
}
```

A complexidade é O(N)

b) Qual o tempo de execução do seguinte algoritmo entre as seguintes alternativas: log N, N, N*logN, N^2

```
double power(double x, int n)
{
 if (n==1) return x;
   else return (power(x,n/2)*power(x,n-n/2));
}
```

A complexidade é O(N*log N)

c) Assuma que lhe é dado uma tabela que contem **N** números pares em que alguns podem aparecer repetidos. Pretende-se determinar se existe um número **K**, que seja igual ao produto de dois números dessa tabela (Se a tabela contiver os números 2,4,6,10, e K=24 então a resposta é positiva e os números são 4,6). Elabore um algoritmo de complexidade $O(N^2)$ para resolução do problema.

```
bool existe( int vec[], int n, int k)
{
   for (int i=0; i<N-1; i++)
     for (int j=1; j<N; j++)
         if (i*j==K)
           return true;
   return false;
}
```

2º Questão (4 Valores)

Considere uma árvore binária tal que a informação contida em cada nó da árvore é composta por um número inteiro e dois apontadores, respectivamente para os filhos direito e esquerdo desse nó. Assuma que cada nó da árvore é do tipo TreeNode, indicado à direita.

```
typedef struct _TreeNode {
 int value;
 struct _TreeNode *right, *left;
}TreeNode;
```

a) Implemente uma função que receba um apontador para a raiz de uma árvore binária com nós do tipo indicado, e que retorne a soma dos valores de todas as folhas da árvore. Utilize a seguinte assinatura para a função: int sum_leafs(TreeNode* treeRoot);

```
int sum_leafs(TreeNode* treeRoot)
{
    if (treeRoot ==NULL)
      return 0;
   else if ((treeRoot->right==NULL) && (treeRoot->left==NULL))
      return theRoot->value;
   else return sum_leafs(theRoot->right) + sum_leafs(theRoot->left);
}
```

b) Supondo que a árvore a que o seu algoritmo é aplicado possui N nós qual a complexidade do código desenvolvido em a) Justifique.

A complexidade é O(N) pois todos os nós da árvore são visitados.

3º Questão (4 Valores)

A direção geral de viação, sendo responsável pelo bom funcionamento rodoviário, tem também como funções manter uma base de dados atualizada com informação respeitante a condutores e veículos. Cada condutor tem associada uma carta de condução, identificada por número, entidade emissora, local de emissão, data de emissão, data de renovação obrigatória e classes de veículos que o condutor está autorizado a conduzir. Cada condutor possui, ainda, uma morada e um cadastro de infrações, com data, gravidade e descrição da infração. Para os veículos, identificados univocamente pela sua matrícula, encontra-se arquivada informação acerca da marca, modelo, cor, ano de fabrico, cilindrada e situação do veículo (normal, abatido ou roubado). Pretende-se um sistema que permita efetuar as seguintes operações:

- Inserir um condutor ou um veículo;
- Remover um condutor (dado o número de carta) ou um veículo (dada a matricula);
- Inserir uma infração no cadastro do condutor (dado o número de carta);
- Listar por ordem crescente de data, a data, gravidade e descrição das infrações de um condutor (dado o número de carta) entre duas datas;
- Listar por ordem crescente de ano de fabrico, toda a informação referente aos veículos numa determinada situação.

Assuma que o número total de infrações de um condutor é, geralmente inferior a 20, e que o número de condutores e de veículos é, à partida, ilimitado.
Explicite detalhadamente as estruturas de dados mais adequadas para implementar estes sistema e descreva sumariamente como funcionariam as várias operações. Indique quais os elementos que residiriam em memória secundária e aqueles que seriam colocados em memória principal.

Como o número de condutores e veículos é muito grande terão de ficar guardados em memória secundária. A utilização de árvores B+ é uma boa opção pois estas permitem reduzir as operações de leitura e escrita em memória secundária, as quais são as mais demoradas num sistema computacional.

Árvore B+ para representar os condutores em que a chave é a sua carta de condução.
Árvore B+ para representar os veículos em que a chave é a matrícula.

- Em cada folha que representa o condutor havia um apontador para uma lista ligada de infrações com sentinela. Esta possui um apontador para o nó que está à cabeça e outro para o nó que está à cauda. A lista está ordenada por data e as novas infrações são sempre inseridas na cauda da lista. A complexidade é O(1) pois a lista possui sentinela.

Existência de 3 listas, uma para veículos normais, outra para abatidos e outra para roubados. Estas listas estão ordenadas por ano de fabrico e possuem apontadores para as folhas da árvore B+ que representam o veículo.

Operações a realizar.

- Inserção de condutor

Inserção de um elemento numa árvore B+ e criação de uma lista vazia de infrações.

- Inserção de veículo

Inserção de um elemento numa árvore B+ e criação de um apontador na lista de veículos normais

- Remoção de um condutor

Remoção de um elemento numa árvore B+ e eliminação da lista de infrações correspondente

- Remoção de um veículo

Remoção de um elemento numa árvore B+ e destruição do apontador na lista respectiva (normal, roubado ou abatido)

- Inserir uma infração no cadastro do condutor (dado o número de carta);

Inserção à cauda de uma infração na lista de infrações correspondente ao condutor

- Listar por ordem crescente de data, a data, gravidade e descrição das infrações de um condutor (dado o número de carta) entre duas datas;

Carregar para memória principal a lista de infrações e percorre-la até encontrar a data inicial começando a listar sequencialmente e terminando na data final.

- Listar por ordem crescente de ano de fabrico, toda a informação referente aos veículos numa determinada situação.

Com base na lista de veículos normais, abatidos ou roubados a qual já está ordenada por ordem de fabrico obter os apontadores para a árvore B+ e listar os dados dos veículos

4º Questão (4 Valores)

Considere à sua direita a definição de uma classe genérica para pilhas (**stacks**) implementadas com vectores:

```
template<class T, unsigned DIM>
class Stack {
    T data[DIM];
    unsigned size;
public:
    Stack() { size=0; }
    bool empty() const { return size==0; }
    T& top();
    void push(const T &p);
    void pop();
};
```

a) Implemente um método pushSpecial(), este método retira o elemento que está no topo da pilha, e avalia o seu valor, de seguida retira da pilha um número de elementos necessários para perfazer o valor indicado no topo da pilha. Caso não seja possível a pilha fica vazia. (Ex: suponha que no topo da pilha está o elemento 5, neste caso vão ser retirados da pilha o topo da mesma e mais 4 elementos, até perfazer um total de 5 elementos).

```
void pushSpecial()
{
   If (!empty())
   {
    T topo = pop();
    int i = topo;
    i--;
    while (! Empty() && i>0)
      pop();
   }
     return;
}
```

b) Mostre a evolução da pilha após as seguintes operações:

```
Stack<int 10> st; int x=3; int z=2; int k=8; int w=9; push(k); push(x); push(z); pushSpecial(); pop(); push(w);
```

1) Pilha = vazia (inicio)
2) Pilha = 8 (após push(k))
3) Pilha = 3 8 (após push(x))
4) Pilha = 2 3 8 (após push(z))
5) Pilha = 8 (após pushSpecial())
6) Pilha = vazia (após pop())
7) Pilha = 9 (após push(w))

5º Questão (4 Valores)

a) Considere o seguinte vetor de inteiros: 0 6 3 2 7 9 5 4 8 5. Represente os passos seguidos para ordenação do vetor utilizando o algoritmo Insertion Sort (ordenação por inserção).

[0 6 3 2 7 9 5 4 8 5]
[0 6 3 2 7 9 5 4 8 5] Final da 1º iteração
[0 3 6 2 7 9 5 4 8 5] Final da 2º iteração
[0 2 3 6 7 9 5 4 8 5] Final da 3º iteração
[0 2 3 6 7 9 5 4 8 5] Final da 4º iteração
[0 2 3 6 7 9 5 4 8 5] Final da 5º iteração
[0 2 3 5 6 7 9 4 8 5] Final da 6º iteração
[0 2 3 4 5 6 7 9 8 5] Final da 7º iteração
[0 2 3 4 5 6 7 9 8 5] Final da 8º iteração
[0 2 3 4 5 5 6 7 8 9] Final da 9º iteração

b) Suponha que se cria uma árvore B^+ de ordem **m** e capacidade **b**, vazia, na qual se inserem os elementos 1,2,3,4,5,6,….k, por esta ordem, com **k** de uma ordem de grandeza muito superior à de **m** e de **b**. Caraterize, de forma precisa e concisa, o estado de ocupação das folhas da árvore, após a inserção dos **k** elementos, e explique claramente a razão desse estado.

Uma árvore B+ de ordem **m** e capacidade **b** tem as seguintes características:

A raiz ou é folha ou possui entre 2 e **m** filhos; Cada nó interno à exceção da raiz tem entre **m/2** e **m** filhos; O número de elementos de cada folha varia entre **b/2** e **b**. Todas as folhas estão ao mesmo nível. **m-1** chaves.

Após a inserção sequencial dos valores 1,2,3,4,…,k. Como k é muito maior que **m** e **b**, a árvore tendera a ter as folhas todas preenchidas da seguinte forma:

A folha mais à esquerda terá os números 1..b
A irmã terá os números b+1..2b e assim sucessivamente.

Ex: Árvore B+ de ordem 4 e capacidade 3